RECEIVED
CENTRAL FAX CENTER
OCT 05 2006

REMARKS

I. Introduction

In response to the Office Action dated July 5, 2006, please consider the following remarks. Claims 1-8, 10-21, and 23-32 are in the application. Re-examination and re-consideration of the application, as amended, is requested.

II. Office Action Double Patenting Rejection

In paragraph (6), the Office Action rejects claims 1-9, 11-21, and 23-31 under the judicially-created doctrine of double patenting as being unpatentable over claims 1-30 of U.S. Patent No. 6,272,521 B1 since the claims, if allowed, would improperly extend the "right to exclude" already granted in the patent.

The Applicants respectfully traverse all of these rejections, but will file a terminal disclaimer if necessary to moot this rejection when allowable subject matter is identified.

III. Office Action Subject Matter Rejection

In paragraph (3), the Office Action rejects claims 1, 13, 19, 24, 28, 29, and 30 under 35 U.S.C. § 101 being the claimed invention is not supported by either a specific and substantial asserted utility or a well established utility.

Applicants believe that the claims clearly describe statutory subject matter with substantial utility, and therefore traverse. Should issues still remain in this regard, the Applicants requests that the Examiner indicate how the rejection can be overcome and how problems may be resolved, in accordance with the directives of the Examination Guidelines for Computer-Related Inventions. See Guidelines II M.P.E.P. § 2106. Specifically, should it be necessary, the Applicants request that the Examiner identify features of the invention that would render the claimed subject matter statutory if recited in the claim. See Guidelines IV, M.P.E.P. § 2106.

The Office Action also asserts that:

> "Claims 1, 19 30 discloses the RX or Write module, claims 13, 24, 29 discloses the Read or TX module. Claims 28 claimed both the Read-Write modules. It's vague, ambiguous, and inconsistent."

Respectfully, the Applicants do not understand this rejection, as it refers to language that is not in the claims. Further, the rationale for the rejection of claim 28 is not provided. Accordingly, the Applicants respectfully traverse.

IV. Non-Art Rejections

In paragraph (4), the Office Action rejected claims 1, 13, 19, 24, 29, and 30 under 35 U.S.C. §112, first paragraph. The Applicants respectfully traverse.

First, since the basis for this rejection appears to be the rejection under 35 U.S.C. § 101 addressed above. the Applicants traverse for the same reasons.

Second, the Applicants do not understand how a rejection under 35 U.S.C. § 101, by itself, supports a rejection under 35 U.S.C. § 112, first paragraph, nor has the Office Action indicated as such.

Third, the all of the claimed subject matter is disclosed and discussed in the Applicants' specification so as to enable one of ordinary skill in the art to have made and used the claimed invention. If the Office Action more specifically indicates which elements of the Applicants claims are at issue, the Applicant would gladly indicate where the requisite teaching can be found in the specification.

Finally, the Office Action suggests that the reason for this rejection is because claims 1, 19, and 30 disclose "only the RX or write module" while "claims 13, 24, 29 discloses the Read or TX module." Again, this recites features that are not in the claims, so the Applicants are unsure how to respond. The argument appears to be that it is impermissible to present claims to a writing operation or a reading operation by themselves. If this is the substance of the intended rejection, the Applicant respectfully traverses, and requests where the basis for such rejection can be found in the MPEP.

In paragraph (5), the Office Action rejected claims 1, 13, 19, 24, 28, 29, and 30 under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicants regard as the invention.

According to the Office Action, these claims are indefinite because they recite a first version without a second version. As the subject claims recite "subsequent" versions in addition to the "first version," the Applicants respectfully traverse.

V. The Cited References and the Subject Invention

A. The Wilcox Reference

U.S. Patent No. 5,568,639, issued October 22, 1996 to Wilcox et al. disclose a method and apparatus for providing an object-oriented file structuring system on a computer. The disclosure provides a method of defining DATA objects and CONTAINER objects and SYSTEM objects that facilitates navigation through a file structuring system. Notation and nomenclature are defined for building files composed of CONTAINERs and DATA and SYSTEM objects and for defining relationships between and among files, CONTAINERs and DATA. A FOCUS LIST tracks objects of interest and aids in NAVIGATION. CONTAINER objects contain other objects. DATA objects enclose DATA in either machine-dependent or machine-independent value representations. Developers work with logical files and can freely create and modify the logical relationships of file objects. A RECONSTITUTION algorithm periodically updates the physical file to correspond to the logical file.

B. The Williams Reference

U.S. Patent No. 5,467,472, issued November 14, 1995 to Williams et al. disclose a method and system for generating and maintaining property sets with unique format identifiers. In a preferred embodiment, a property set stream is generated. The stream comprises three parts: a header, a section locator array, and one or more sections. The header contains information for uniquely identifying the property set and for identifying the number of sections within the property set. The section locator array contains a unique identifier for each section and an offset indicating where the section resides within the stream. The third part, the section definitions, contains the information necessary to maintain groups of properties for each section. Each section contains a section header, a property locator array, and an array of property type/value pairs. The section header indicates both the size of the section and the number of properties defined within the section. The property locator array contains unique property identifiers for each property defined in the section and a relative offset, from the beginning of the section, to the property definition. Each property definition appears as a type/value pair, the type indicator indicating the data format for the property and the value field containing or referencing the data. In a preferred embodiment, a property set is generated by allocating appropriate storage and by storing values in the standard structure for a property set.

VI. Office Action Prior Art Rejections

In paragraph (7), the Office Action rejected claims 1-8, 10-21, and 23-31 under 35 U.S.C. § 103(a) as unpatentable over Wilcox et al., U.S. Patent No. 5,568,639 (Wilcox) in view of Williams et al., U.S. Patent No. 5,467,472 (Williams). Applicants respectfully traverse these rejections.

With Respect to Claims 1, 19, and 29: Claim 1 recites:

> *A method of transmitting a data segment in a data stream using a write module which implements a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, the method comprising the steps of:*
>
> *(a) outputting a first stream of data according to a first version of the streaming protocol;*
>
> *(b) sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol; and*
>
> *(c) delimiting the data segment in the data stream using begin and end tags.*

According to the Office Action, Wilcox discloses a write module that implements a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to the previous version as follows:

> Well-known file formats that are suitable for the storage of loosely-structured data for the purpose of data transfer include SDF (system data format), which represents data units as ASCII strings terminated by linefeed characters; ANSI X.12 standard, which defines several layers of data units and sub-units by a system of in-stream specific-value separator bytes and type values which key to externally documented schemas; the telegraphy standard, which uses in-stream control values to describe the rough layout of data units within the stream; and the BASIC language convention of using comma-delimited ASCII data fields, generally in the context of a schema. In this category of file formats, emphasis is upon the containment and identification of data units independent of data management facilities; thus, access to data held in such formats remains a direct file access problem.

The foregoing discloses a telegraphy standard that is used for storing loosely stored data using in-stream control values to describe the rough layout of data units within the stream. It does not describe or mention different protocols of that stream, nor that such protocols are additive to previous protocol versions.

The Office Action also refers to this portion of the Wilcox reference:

> Once generated an object has a TYPE defined at its generation, and thereafter it can be managed only as defined in the invention for objects of that TYPE. Thus, for example, data written as an INTEGER DATA object can only be read as an integer, and any attempt to read it as another DATA type will fail and any attempt to move into it (which a CONTAINER allows) will fail. Likewise, any attempt to read the value of a CONTAINER will fail, but movement into the CONTAINER will succeed.

The Applicants are unsure of how the foregoing passage is relevant. The Office Action also notes that the following refers to an "earlier version":

> The JOURNAL can be used together with a base file (a copy of an earlier version of the MFILE) to reconstruct the MFILE in case of a computer malfunction. Similarly, the TAIL of an MFILE is used to reconstitute an MFILE in the event of a computer malfunction.

This refers to an earlier version of a file, not an earlier version of a protocol, as recited in claim 1.

The Office Action does not mention where the step of "outputting a first stream of data according to a first version of the streaming protocol" can be found. Wilcox does not disclose different versions of streaming protocols.

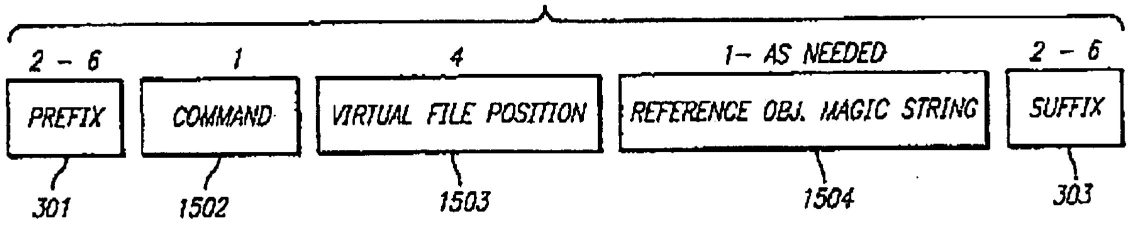
With regard to the step of "sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol", the Office Action refers to the following portion of the Wilcox reference:

The 'Reference Object Magic String' 1504 in FIG. 15A is a variable-length numeric sequence formatted like a numeric Dewey Decimal number, which represents the logical position of the reference object in the file. The section above on Symbolic Notation uses a printable representation of this value to indicate the logical positions of objects in the diagrams shown. The actual encoding of the 'Magic String' is explained below in the section on the Object Control Scheme. The 'Magic String' for the Reference Object is obtained from the FOCUS ENTRY for the current FOCUS when a HANDLE COMMITs an alteration.

The sequentially-appended SYSTEM objects preserve a copy of the MAGIC STRING of the reference object that was held in FOCUS by a HANDLE when the HANDLE made the change. Thus, if the SYSTEM objects are needed to reconstruct the FOCUS LIST that contains the FOCUS ENTRIES, each SYSTEM OBJECT can refer to an already-recreated FOCUS ENTRY that can be located by its unique MAGIC STRING in the FOCUS LIST, and the subsequent updating of the FOCUS ENTRY logical position by other file alterations can thereafter occur in memory.

The foregoing refers to a system object. The system object can be appended to add new objects, and objects can be inserted into or removed from the system object. This is described in column 17, lines 20-39). The structure of the system object is shown in FIG. 15A below:

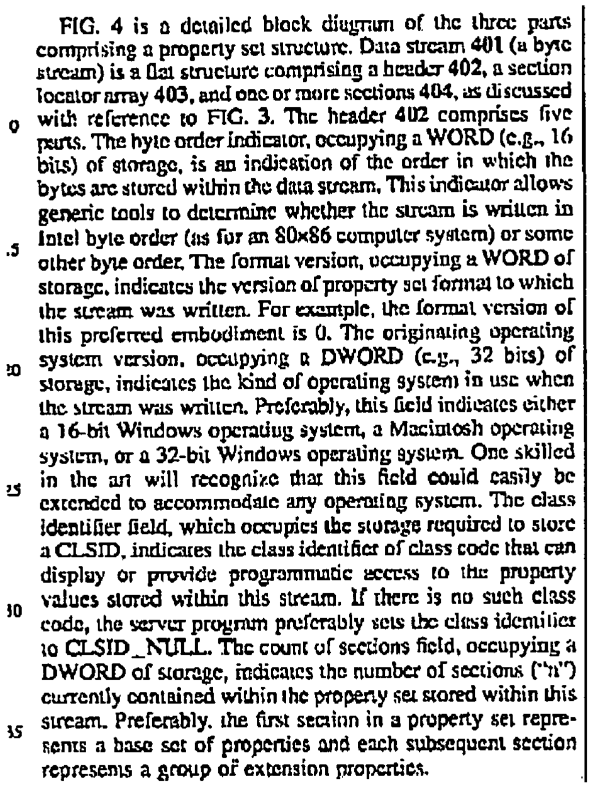
FIG. 15A



However while the foregoing discloses appending an object with other objects (and hence data), it does not disclose "according to each subsequent version of the streaming protocol up to an including the selected version."

Further, it does not disclose doing so "if the selected version of the streaming protocol is not the first version of the streaming protocol," as recited in claim 1.

The Office Action acknowledges this, but relies on Williams for this feature. According to the Office Action, Williams discloses a method and system for generating and maintaining property sets with unique format IDs including the originating version as follows (the Office Action also refers to col. 36, lines 5-36, but there is no column 36):



> FIG. 4 is a detailed block diagram of the three parts comprising a property set structure. Data stream 401 (a byte stream) is a flat structure comprising a header 402, a section locator array 403, and one or more sections 404, as discussed with reference to FIG. 3. The header 402 comprises five parts. The byte order indicator, occupying a WORD (e.g., 16 bits) of storage, is an indication of the order in which the bytes are stored within the data stream. This indicator allows generic tools to determine whether the stream is written in Intel byte order (as for an 80x86 computer system) or some other byte order. The format version, occupying a WORD of storage, indicates the version of property set format to which the stream was written. For example, the format version of this preferred embodiment is 0. The originating operating system version, occupying a DWORD (e.g., 32 bits) of storage, indicates the kind of operating system in use when the stream was written. Preferably, this field indicates either a 16-bit Windows operating system, a Macintosh operating system, or a 32-bit Windows operating system. One skilled in the art will recognize that this field could easily be extended to accommodate any operating system. The class identifier field, which occupies the storage required to store a CLSID, indicates the class identifier of class code that can display or provide programmatic access to the property values stored within this stream. If there is no such class code, the server program preferably sets the class identifier to CLSID_NULL. The count of sections field, occupying a DWORD of storage, indicates the number of sections ("n") currently contained within the property set stored within this stream. Preferably, the first section in a property set represents a base set of properties and each subsequent section represents a group of extension properties.

However, the foregoing says does not disclose conditioning any kind of action on a selected version of a streaming protocol. The Applicants are confused as to how the foregoing portion is in any way relevant to claim 1. At best, the Williams reference merely discloses the use of format identifiers. Nor do the Applicants understand the proffered teaching for combining the references, as Wilcox and Williams appear to be capable of "controlling and maintaining a relationship" between versions. The Applicants are at a loss to understand what would be gained by modifying either of the references to that of the other.

Referring back to the step of step of "sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol", the Office Action also alleges that this feature is disclosed in Wilcox as follows:

> Every additive alteration (insert, replace, append) creates a new object. When an additive alteration action is invoked, the system (1) creates a receiving buffer; (2) marks the buffer to indicate the alteration type; (3) receives the DATA written into the buffer; and (4) removes or posts the buffer to a serialized UPDATES QUEUE, where the contents of the buffer appear as a SYSTEM object followed by the added object. In case of 'remove' the invention posts to the UPDATES QUEUE only the information required to build a SYSTEM object.
>
> In the preferred embodiment the intended alteration must be declared through the HANDLE before any content is written to the receiving buffer, and the successful completion of the update must be accomplished by a COMMIT action. The COMMIT action causes the receiving buffer to be posted to the UPDATES QUEUE. An alteration pending in this sequence may also be ABANDONED, in which case the receiving buffer is erased and nothing is posted to the UPDATES QUEUE. An ABANDON or COMMIT action terminates the alteration.

and

> The sequentially-appended SYSTEM objects preserve a copy of the MAGIC STRING of the reference object that was held in FOCUS by a HANDLE when the HANDLE made the change. Thus, if the SYSTEM objects are needed to reconstruct the FOCUS LIST that contains the FOCUS ENTRIES, each SYSTEM OBJECT can refer to an already-recreated FOCUS ENTRY that can be located by its unique MAGIC STRING in the FOCUS LIST, and the subsequent updating of the FOCUS ENTRY logical position by other file alterations can thereafter occur in memory.

and

Once the placement of an alteration has been established by the CURRENT FOCUS and the selected alteration, the action required to add an object in an alteration depends upon the alteration selected and type of object that is to be generated. The full sequence of actions to add a DATA object is ALTER, WRITE, COMMIT; the full sequence required to add a CONTAINER to the MFILE is ALTER, CREATE CONTAINER, COMMIT (in this example the added container will be empty). In these sequences the word ALTER stands for one of INSERT, REPLACE and APPEND. The general concept is that the placement of the alteration is announced in advance with reference to the current FOCUS. The concepts expressed by the action phrases given above may be embodied in differing function names without changing the invention.

and finally,

Identifier. When the 'Identifier' flag is set, the 'Identifier' field contains a 64-bit number that is unique to the object. Each computer on which the invention is used is set up with a unique node identifier in a configuration file to assure universal uniqueness of the identifier, because the node identifier number provides part of the 64 bits of value. The invention maintains its own unique object sequence numbering that is applied to the other bits. This numbering system facilitates the association of objects with one another by programs which use the invention.

But as best the Applicants can determine, the foregoing discloses appending information from a file, but not "sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version." Indeed, there is no notion at all of addressing the problem of different streaming protocols.

Finally, the Office Action suggest that the following discloses "delimiting the data segment in the data stream using begin and end tags."

> The BRACKETs of an object are identical in size, and that size is encoded in identical KEYBYTEs that appear as the first and last bytes of the object. Each BRACKET encodes the offset at which the KEYBYTE of an adjacent object might be found: this offset is found in the PREFIX for 'next' objects and in the SUFFIX for 'prior' objects. The absolute values of these offsets are identical. Using this scheme, objects of varying size can be included in the file without reserving the space that would be required for fixed-length objects and without overloading the range of directly representable values in order to use a delimiting symbol.

and

> HANDLEs can be declared to exist in a program, each of which is open upon an MFILE and any number of which may be open on the same MFILE. At creation of a HANDLE, its focus is on the HEAD of an MFILE, which is itself a CONTAINER object.
>
> In concept, a HANDLE navigates from object-to-object in an MFILE by jumping forward the distance encoded in the PREFIX of the current object for forward traversing or jumping backward the distance encoded in the SUFFIX of a current object for backward traversing. This concept holds true only for the ideal MFILE that has only a HEAD, because in that state all objects have the same physical and logical order. Because it is necessary for HANDLEs to be able to navigate at all times, however, the invention relies on indirect navigation by causing the HANDLE to reference FOCUS ENTRIES in the sorted order in which they appear in the FOCUS LIST. The invention creates from the physical file object a new FOCUS ENTRY whenever one is needed that is not present as part of the FOCUS LIST in memory.

But the foregoing does not appear to disclose the use of beginning and end tags to delimit a data segment in a data stream. Wilcox deals with the storage and retrieval of files, not the transmission of data streams.

BEST AVAILABLE COPY

> The present invention causes each file to be altered only by additions to the tail of the file, so that the physical location of previously-added DATA is stable and thus easily accessible in a multi-threaded environment. Information is retained in the FOCUS LIST and in the TAIL of the MFILE to map the location of removed objects which have not yet been removed physically from the MFILE and to map the logical positioning of all objects added to the TAIL of the MFILE which are not yet in physical positions which match their logical positions. Placement of alterations at the end of the MFILE facilitates multi-thread multi-handle file access and allows multiple threads of execution to access, read, and modify various parts of a single file at the same time.

Wilcox is directed to an entirely different problem than the Applicant's invention, that of storing data objects in a way so that they may be used concurrently by multiple processing threads: As such, Wilcox discloses rather complex techniques by which the data within each file can be directly obtained by the thread that is using the data. While this certainly appears to be a useful result, it has little to do with the Applicants invention, which addresses the problem of forward and backward compatibility among streaming protocols. Simply put, Wilcox is only remotely related to the Applicant's invention, if at all.

Accordingly, claim 1 is patentable over the prior art of record, and the Applicants respectfully traverse this rejection. Claims 19 and 29 are patentable for the same reasons.

With Respect to Claim 13, 24, and 30: Claim 13 recites a method for receiving a data segment prepared according to the features of claim 1. Specifically:

> *A method of receiving a data segment from a data stream using a read module which implements a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, the method comprising the steps of:*
>
> *(a) receiving a first stream of data according to a first version of the streaming protocol;*
>
> *(b) if the selected version of the streaming protocol is not the first version of the streaming protocol, sequentially receiving additional streams of data according to each subsequent version of the streaming protocol up to and including the selected version; and*
>
> *(c) testing, prior to receiving each additional stream of data, whether an end of the data segment has been detected, and if so, terminating reception of the data segment prior to receiving the additional stream of data according to the selected version.*

The Office Action argues that claim 13's step of "testing, prior to receiving each additional stream of data, whether an end of the data segment has been detected, and if so, terminating reception of the data segment prior to receiving the additional stream of data according to the selected version" is disclosed as follows:

> The PREFIX and SUFFIX structures of an object encode the object category (CONTAINER, SYSTEM, or DATA), the size of the object, and additional information appropriate to the category. Any object of each of the types identified in this paragraph, and all of their subtypes, can be type-identified from the PREFIX or SUFFIX structure of the object. The invention includes a facility for defining further sub-types of SYSTEM, CONTAINER and DATA categories.

and

> The encoding described in the preceding paragraph is illustrated in FIGS. 16A, 16B and 16C. FIG. 16A shows a MAGIC STRING that represents an object nested inside a CONTAINER, in which the MAGIC STRING has 3 components: a byte series for level 1, a byte series for level 2, and a null terminating byte. A byte series consists of 1 or more bytes. FIG. 16B illustrates a byte series consisting of 3 bytes, of which the value bits are represented by letters A, B and C, and the terminating flag bit is represented by the number '1'. FIG. 16C shows the result of packing the value bits together into a 32-bit variable, left-padded with '0' bits, which yields an integer number that gives the cardinal position of the object at the represented level.

The foregoing discloses PREFIX and SUFFIX structures which encode the category of the objects, and magic strings, which represent objects nested inside a "container." Respectfully, the Applicants do not understand how this discloses the features of claim 13. The Applicants therefore traverse the rejection of this claim as well. Claims 24 and 30 are patentable for the same reasons.

With Respect to Claim 28: Claim 28 recites both the writing and reading functions of the claims discussed above, and is patentable for the same reasons.

VII. Dependent Claims

Dependent claims 2-8, 10-12, 14-18, 20-21, 23, 25-27, 31, and 32 incorporate the limitations of their related independent claims, and are therefore patentable on this basis. In addition, these claims recite novel elements even more remote from the cited references. Accordingly, the Applicant respectfully requests that these claims be allowed as well.

VIII. Claim 32

The patentability of claim 32 was not addressed in the Office Action. The Applicants respectfully request that the patentability of this claim be addressed in the next communication.

IX. Conclusion

In view of the above, it is submitted that this application is now in good order for allowance and such allowance is respectfully solicited. Should the Examiner believe minor matters still remain that can be resolved in a telephone interview, the Examiner is urged to call Applicants' undersigned attorney.

Respectfully submitted,

GATES & COOPER LLP
Attorneys for Applicant(s)

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: October 5, 2006

By: [illegible]
Name: Victor G. Cooper
Reg. No.: 39,641

VGC/

G&C 30571.77-US-01